1. Data Pre-processing : Load a dataset, handle missing value encode categorical data and normalise / standardise feature ?

```
import pandas as pd
import numpy as np
from sklearn . impute import SimpleImputer
from sklearn . preprocessing import onetlot Encoder . StandardScaler
from sklearn . Compose import Column Transformer
from sklearn . pipeline import pipeline

#Sample dataset
data = pd. DataFrame ({
'Age' : [25, np.nan , 35, 40, 29],
'Salary' : [50000, 60000, np.nan , 80000, 52000],
'Department' : ['sales', 'Engineering', 'HR', np.nan, 'Sales'],
'purchased' : ['Yes', 'No', 'Yes', 'No', 'Yes']
})

#Display original dataset
print("original Dataset: \n")
print (data)

# separate feature and target
x = data. drop ('Purchased', axis =1)
Y= data ['Purchased']
```

```
#identify numerical and categorical columns
numerical_cols = x.select_dtypes(include=['int64','float64']).
  columns.tolist()
categorical_cols = x.select_dtypes(include=['object']).columns.tolist()
#defines preprocessing for numerical data (imputation + standardization)
numerical_pipeline = Pipeline(steps=[('imputer', SimpleImputer(strategy='mean')),
('scaler', StandardScaler())
])
# Define preprocessing for categorical data (imputation + one-hot encoding)
categorical_pipeline = Pipeline(steps=[('imputer', SimpleImputer(strategy=
'most_frequent')), ('encoder', OneHotEncoder(handle_unknown='ignore')]
])
#combine preprocessing
preprocessor = ColumnTransformer(transformers=[('num', numerical_pipel
  numerical_cols),
('cat', categorical_pipeline, categorical_cols)
])
#Fit and transform the features
X_processed = preprocessor.fit_transform(X)
#Display processed features
print("\n Processed Features (after handling missing values, encoding,
  and scaling):\n")
print(X_processed.toarray() if hassattr(X_processed, "toarray")
  else X_processed)
```

```
#optionally show the transformed feature names encoded_feature_name
= preprocessor.named_transformers_['cat']['encoder'].get_feature_
names_out(categorical_cols) all_feature_names = numerical_cols+
encoded_feature_names.tolist()
print("\n Processed Feature names:")
print(all_feature_names)
```

1st program

```
Original Dataset:

   Age  Salary   Department Purchased
0  25.0  50000.0        Sales       Yes
1   NaN  60000.0  Engineering        No
2  35.0      NaN           HR       Yes
3  40.0  80000.0          NaN        No
4  29.0  52000.0        Sales       Yes

Processed Features (after handling missing values, encoding, and scaling):

[[-1.41775817 -0.98950981  0.          0.          1.        ]
 [ 0.         -0.04711951  1.          0.          0.        ]
 [ 0.53777034  0.          0.          1.          0.        ]
 [ 1.5155346   1.83766107  0.          0.          1.        ]
 [-0.63554677 -0.80103175  0.          0.          1.        ]]

Processed Feature Names:
['Age', 'Salary', 'Department_Engineering', 'Department_HR', 'Department_Sales']
```